2003-12-08

An Editor for Adaptive XML-Based Policy Management of IPSEC

Mohan, Raj

Computer Security Applications Conference (ACSAC)

Annual Computer Security Applications Conference (ACSAC), December 8-12, 2003, Las

An Editor for Adaptive XML-Based Policy Management of IPsec

Raj Mohan
Indian Army
rajmohan_iyer@hotmail.com

Timothy E. Levin
Naval Postgraduate School
levin@nps.navy.mil

Cynthia E. Irvine
Naval Postgraduate School
irvine@nps.navy.mil

Abstract

The IPsec protocol provides a mechanism to enforce a range of security services for both confidentiality and integrity, enabling secure transmission of information across networks. Dynamic parameterization of IPsec, via the KeyNote trust management system, further enables security mechanisms to adjust the level of security service "on-the-fly" to respond to changing network and operational conditions. However KeyNote requires that an IPsec policy be defined in the KeyNote specification syntax. Defining such a dynamic security policy in the KeyNote Policy Specification language is complicated and can lead to incorrect specification of the desired policy, thus degrading the security of the network. We present an alternative XML representation of this language and a graphical user interface to create and manage a consistent and correct security policy. The interface has the simplicity of a simple menu-driven editor that not only provides KeyNote with a policy in the specified syntax but also integrates techniques to support administrative policy verification.

1. Introduction

1.1. Objective

Network Protocols such as IPsec and trust management systems like KeyNote provide mechanisms to secure computer-to-computer communications. These tools enable the user to use various encryption and authentication mechanisms to ensure confidentiality, integrity and non-repudiation of communications. Dynamic parameterization [1] of IPsec further enables security mechanisms to adjust the level of security service "on-the-fly" to respond to changing network and operational conditions. The *trust management system*, KeyNote, specifies a language for describing actions, which are operations with security consequences that are to be controlled by the system [3][4][5]. The language also provides the syntax for specifying the application policies, which govern the actions that the *principals* are authorized to perform. To translate a dynamic organizational security policy into the KeyNote specification language is, however, a daunting task due to the complexities of the language and the policy. An incorrect specification of the security policy might result in a compromise of network security. It is in this context that the need for an alternative policy specification mechanism is identified. This mechanism should enable the user to clearly and correctly specify the policy and also support user verification that the specified policy is free of inconsistencies and contradictions. The purpose of this work is to analyze, design and implement a policy editor interface that guides a user to specify various attributes of the IPsec security policy. The policy is stored in an intermediate XML format. The program will automatically generate the equivalent policy in the KeyNote specification language. A presentation mechanism will be described for providing the user with an intuitive view of the policy that can be helpful in preventing inconsistencies and contradictions in the specified policy.

1.2. Background

The increased dependence on computers for communication has enhanced the importance of network security. The use of an inherently insecure Internet as the medium for communicating sensitive material requires that the end users have capabilities to ensure that the data transmitted is secure. Furthermore, network administrators should have means to translate the desired organizational security policy into an automated security policy and have mechanisms to implement this policy over their network.

IPsec [10] extends the IP Protocol to enable security for TCP/IP communications. IPsec provides both secrecy and integrity services. A wide variety of choices are available when establishing protected communications across the network. The appropriate choice and combination of secrecy and integrity mechanisms will depend upon the many-faceted "trust relationships" between the communicating entities and the security environment. Those relationships are constrained by the policy of each entity. Negotiation of policy and mechanisms takes place in the context of the Internet Key Exchange (IKE) framework and the Internet Security Association and Key Management Protocol (ISAKMP)

[11]. However, IKE and ISAKMPD do not provide a general mechanism for managing and incorporating security policy. In order to ensure that IPsec consistently meets the local security policy needs of the user, a trust management system may be used to encode policy and support communications security negotiation and management [15].

A trust management system unifies the elements of security policy, credentials, access control, and authorization. The IPsec implementation in OpenBSD utilizes a trust management system to manage security according to policy [2]. Applications can use the KeyNote trust management system to verify, through the compliance checker, whether a requested policy addition or change is authorized [6].

The Quality of Security Service (QoSS) model provides a means to manage security services based on the requirements set by the user's requests, the system's security policy, the availability of system resources and the network environment [8]

Dynamic parameterization of IPsec [1] via enhancements to KeyNote provides more granularity in IPsec and provides flexibility to adjust security controls according to changes in threat conditions, critical time transmissions, and network congestion/traffic. This makes IPsec a QoSS mechanism.

The effectiveness of this mechanism depends on having in place a correct security policy specified in the KeyNote specification language. For practical real-life network operations, specifying such a dynamic and granular policy is an almost insurmountable task due to the syntactic complexity of the KeyNote language and the inherent complexity of the policy logic involved. An XML-based specification of the policy should provide the desired flexibility, be easy to use, and support an interface for administration of the security policy. Such a multi-function toolkit should provide an abstraction of the KeyNote language and enable users to better utilize the power of IPsec and KeyNote in managing network security [12].

1.3. Expected Benefits

A policy management toolkit will enable critical commercial, government and military computer and communications systems to automate security service adjustments according to dynamic environmental parameter settings, such as current *network security status* (INFOCON and THREATCON, in military parlance). The use of XML in such an effort enables common use of available XML tools for ensuring security policy consistency and also utilizes the flexibility and compatibility that XML provides. An easy-to-use interface ensures its use and the resulting policy correctness will provide confidence in the overall security implementation of the network.

1.4. Organization of this Paper

The paper will be organized as follows: Section 2 provides an overview of the QoSS model. Section 3, KeyNote Support for QoSS, reviews the KeyNote language and its specification for the QoSS implementation in OpenBSD 2.8. Section 4, XML and Policy Representation, describes XML technologies and their application to the problem domain. Section 5, Design and Implementation, presents the design philosophy of the toolkit; the considerations and overall architecture will be discussed in detail, and implementation issues of the components will be highlighted. Section 6 describes future work, and Section 7 summarizes our results.

2. Quality of Security Service (QOSS)

IPsec provides a high degree of granularity in discriminating between traffic that is afforded IPsec protection and traffic that is allowed to bypass IPsec. Further use of a trust management system such as KeyNote enables an application to simply ask the compliance checker whether a requested action should be allowed. We can use KeyNote to specify a granular security policy to be enforced by IPsec and also use KeyNote to verify communications requests based on the policy. Then, we would be able to modulate the KeyNote security policy settings dynamically in accordance with the security and performance requirements of the applications in particular, and networks as a whole. This is the essence of 'Quality of Security Service' (QoSS).

2.1. Dynamic Parameters and Network Modes

Many organizations utilize a variety of dynamic parameters to define a predefined response of specific actions according to policy. For the Government and DoD, examples include INFOCON and THREATCON levels. In order for a security mechanism to be fully functional within such a dynamic infrastructure, it has to be able to reflect those dynamic parameters in its security posture. A change in an INFOCON or THREATCON level should have an immediate effect on attributes and settings in the low-level security mechanisms. Security level and network mode, defined in the following sections, have been chosen as two abstract dynamic parameters that govern changes to security attributes as defined in the organization's security policy [13].

In the examples described here, we use the following network modes: normal, impacted, and crisis. *Normal mode* is defined as ordinary operating conditions with normal traffic load and no heightened threat conditions. *Impacted mode* may be defined when the network/system is experiencing high levels of traffic and therefore certain security selection may not be available due to efficiency constraints. *Emergency mode* may be defined as a

situation that requires the highest level of security or the lowest level dependent on the situation and policy [14].

2.2. User Choices for Security Levels.

Network security policies may utilize a range of maximum and minimum-security levels for each variant security service. Minimum-security levels set the lowest acceptable security attributes and maximum-security levels establish a ceiling on the use of available security resources. Intersections of policies require further granularity in security settings to satisfy all governing users and systems. A user may also desire to select a higher level of security than the predefined minimum.

A user or application, however, may quickly become overwhelmed with the security setting details, such as the specific key and algorithm settings of a cryptographic protocol, potentially resulting in degraded security or performance. By developing security definitions that encompass detailed security settings required by users or applications, the complexity of the selection process for the security settings can be simplified to a reasonable level. Our examples involve the use of the following abstract security levels: *high, medium and low*. 'High' security level would utilize strong levels of security attributes, medium level, moderate level of security attributes, and low level, low to no security attributes. In this approach, the system security resource manager or security engineer is responsible for presetting (mapping) the detailed security variables in accordance with the abstract security levels offered to users or applications.

2.3. Mapping Abstract Parameters to Security Mechanism.

A mapping of abstract dynamic parameters to resident security mechanisms is required to properly enforce policy decisions. For example, network modes may be mapped to security level ranges and ultimately to security attributes and settings, as shown in Table 1.

Table 1. Mapping security policies to security attributes

Network Mode	Security Level	Security Attributes
Normal	Low	Encryption: None Authentication: MD5
	Medium	Encryption: DES Authentication: MD5
	High	Encryption: 3DES Authentication: MD5
Crisis	Low	Encryption: None Authentication: None
	Medium	Encryption: None Authentication: None
	High	Encryption: DES Authentication: MD5
Impacted	Low	Encryption: 3DES Authentication: MD5
	Medium	Encryption: 3DES Authentication: SHA
	High	Encryption: AES Authentication: SHA

The security resource manager and security engineer would define the network modes and security levels to provide the users and applications with appropriate security service. Once defined, the complexity of the security mechanism and security attribute selection is transparent to the user.

2.4. Implementation Issues

Quality of Security Service (QoSS) is a model for the modulation of security settings and enhancement of performance based on both necessity (e.g. threat) and resource availability. It also provides a tool to ensure that the minimum-security requirements of applications and the network as defined in the security policy is not violated. Hence, an adaptive security policy based on network threat and performance conditions is the key to optimal and secure utilization of the network resources. KeyNote provides one such policy specification language but its practical implementation with complex policy statements is extremely difficult. A policy toolkit providing an abstraction for this language was therefore felt necessary. Our plan was to use the power of KeyNote for formal compliance checking and at the same time be easy to use and administer.

3. KeyNote Support for QoSS

The syntax and semantics of the KeyNote language is described in detail in RFC 2704 [3]. The language is used for specifying application 'policies,' which govern the actions that principals (entities that can be authorized to perform actions) are authorized to perform. The language provides the semantics for describing 'actions,' which are operations with security consequences that are to be controlled by the system. It is also used for specifying 'credentials', which allow principals to delegate authorization to other principals.

KeyNote assertions are divided into 'fields' that serve various semantic functions. Each field starts with an identifying label at the beginning of a line, followed by the ":" character and the fields contents.

One mandatory field is required in all assertions:

- Authorizer

Six optional fields may also appear:

- Comment
- Conditions
- KeyNote-Version
- Licensees
- Local-Constants
- Signature

The conditions field is used to define a security policy. This field gives the 'conditions' under which an action may be performed.

Security attributes reside in the conditions field and are expressed in the form of a logical expression. The expression's syntax is similar to that of a programming language "if statement". The expression is usually broken into sub statements by using && ("and"), || ("or"), and parenthesis to construct logical conditions. For example the following phrase describes two security proposals supporting Telnet services (service_port= 23) using ESP with 3DES for encryption and finger services (service_port=79) using AH with SHA for authentication:

```
(local_filter_port == "23" &&
        esp_present == "yes" &&
        esp_enc_alg == "3des") ||
(local_filter_port == "79" &&
        ah_present == "yes" &&
        ah_auth_alg == "sha") -> "true";
```

Using the example above, with security levels "high" and "low" and network modes "normal" and "impacted", the condition expression is expanded:

```
Conditions: ( (app_domain == "IPsec policy") && (
( (network_mode = "normal" &&
((security_level = "high" &&
 ((local_filter_port == "23" &&
        esp_present == "yes" &&
        esp_enc_alg == "3des") ||
(local_filter_port == "79" &&
        ah_present == "yes" &&
        ah_auth_alg == "sha"))) ||
((security_level = "low" &&
 ((local_filter_port == "23" &&
        esp_present == "yes" &&
        esp_enc_alg == "des") ||
(local_filter_port == "79" &&
        ah_present == "yes" &&
        ah_auth_alg == "des-mac")))) ||
(network_mode = "impacted" &&
((security_level = "high" &&
 ((local_filter_port == "23" &&
        esp_present == "yes" &&
        esp_enc_alg == "aes") ||
(local_filter_port == "79" &&
        ah_present == "yes" &&
        ah_auth_alg == "sha"))) ||
((security_level = "low" &&
 ((local_filter_port == "23" &&
        esp_present == "yes" &&
        esp_enc_alg == "3des") ||
(local_filter_port == "79" &&
        ah_present == "yes" &&
        ah_auth_alg == "sha-md5")))) -> "true";
```

This example illustrates that the complexity of the language increases dramatically as we add more ports and parameters to it. The nesting of parenthesis to multiple levels makes writing a syntactically correct policy file difficult. In the following section, the use of XML is described for the practical specification of the KeyNote policy file.

4. XML and Policy Representation

Extensible Markup Language (XML) [16] is a rapidly maturing technology with powerful real-world applications, particularly for the management, display and organization of data. XML is a technology concerned with the description and structuring of data. It is a subset of Standard Generalized Markup Language (SGML), with the same goals, but with much less complexity. XML is not a language but a standard for creating languages that meet the XML criteria. It describes a syntax that you use to create your own languages [7].

Data is separated from presentation in XML. XML structures the data, while style sheets format the data presentation. That makes it easier to use the data for multiple purposes. The same stylesheet can be used with multiple documents to create a similar appearance among them. Alternatively, multiple stylesheets can be applied to an XML document to provide different forms of presentation of the data. There are a variety of languages that can be used to create stylesheets such as Extensible Stylesheet Language Transformations (XSLT).

4.1. XML DTDs and Schemas

XML includes two methods of checking the validity o an XML document: *document type definitions* (DTDs) and *schemas*. A document is valid if its XML content complies with a definition of allowable elements, attributes and other document pieces. By utilizing special 'Document Type Definition' syntaxes or DTDs, it is possible to check the content of a document type with a special parser.

A schema is the XML construct used to represent the data elements, attributes, and their relationships as defined in the data model. By definition, a DTD and a schema are very similar [17]. However, DTDs usually define simple, abstract text relationships, while schemas define more complex and concrete data and application relationships. A DTD doesn't use a hierarchical formation, while a schema uses a hierarchical structure to indicate relationships. XML schema definitions are also commonly referred to as XSD.

4.2. XSLT

XSL [19] is used to create stylesheets. An XSL engine uses these stylesheets to transform XML documents into

other document types, and to format the output. Extensible Stylesheet Language Transformations (XSLT) is a language which can transform XML documents into any text-based format, XML or otherwise. Stylesheets define the layout of the output document and the location of the data in the source document. That is, "retrieve data from this place in the input document; make it look like this in the output". In XSL parlance, the input document is called the source tree, and the output document the result tree.

4.3. Advantages of XML for the Policy Specification Language

As described above we have a need to represent the intended IPsec policy in a form separate from the native KeyNote representation. Some of the advantages that would accrue by using XML are as follows:

4.3.1. **Tools.** Use of XML for specification of the KeyNote policy file lends itself to be used with the freely available, verified, tested and user-friendly tools. These tools include among others, XML editors, parsers, validators, translators etc. The availability of such tools and the extensive use of XML in modern communication protocols and other programs will enable users to manipulate XML files easily. Wide availability of such tools will also help in creating and maintaining the policy files over diverse systems without the need for an application specific editor.

4.3.2. Platform Independence. It is possible to edit, maintain and distribute the XML policy file across different OS platforms.

4.3.3. Single Data Multiple Presentation. Once we represent the policy in an XML format it is possible to extract relevant information and present it in different forms that are more intuitive and useful to the administrator or the user. XSLT style sheets can be written and associated with the policy file to generate different presentation formats. Presenting it in a more understandable, graphic format will help the administrator identify any inaccuracies, inconsistencies, or contradictions in the policy file. Intelligent agents can also be written to audit the policy file and signal the administrator for errors in the policy file.

4.3.4. Consistency and Accuracy. XML schemas and/or DTDs can be used to validate the XML file to see if it matches our specifications. Validating the policy file with a well-defined schema will enable errors to be picked up. This will trap all errors without having to go through the entire file manually. The use of generic schema generators and validators only makes this an easier task. This will also support users in their verification of policy files received across the networks.

4.3.5. Extensible Format. An XML format will allow the extension of the policy file to include new constructs. Additional tags can be defined for elements and attributes as and when the need to incorporate them arises. This would not require changes to the application code as long as the structure of the document is maintained.

4.3.6. Ease of Use. The hierarchical nature of XML layout results in an easy to use and easy to manipulate format. It makes the file more modular and more easily understandable.

4.3.7. Semantic Content Use. The semantic content of the policy file enables future deployment of intelligent agents or roaming agents that can read policy files and report problems, and that can resolve conflicts between multiple systems by highlighting for instance the difference in the policies between them.

4.4. Integrating XML and KeyNote Policy

The KeyNote engine requires that the assertions, credentials and the policy files be specified in its native syntax. As discussed, there is a clear problem of differentiation between XML data content and its representation. Specifying the policy data in an XML format would enable us to use XSL to translate the data to any format needed, such as a more human readable form. Furthermore, specifying an XML schema would provide us the benefit of validating the XML policy file for correctness prior to its transformation to KeyNote syntax.

5. Design and Implementation

The first challenge to using XML for security policy specification was to determine an alternate representation of the policy logic in the form of an XML *user policy file*. The overall approach was to develop a format for the user policy file and then create a style sheet to transform it to the native KeyNote policy file format. This approach also provides the flexibility of later being able to extract useful administrative information from the user policy file.

Arriving at a format for the user policy file is a challenging task and there are multiple options available. The primary requirement is that the resulting XML file be well formed. During this research, multiple formats were considered. Each had its strengths and shortcomings. For instance one format would lend itself to an easy application design while another would permit more semantic content in the file format. The former therefore makes it easier to write an application such as a 'Policy Editor' while the latter results in a more descriptive self-defining file, which could be a good interchange format between multiple applications. However we realized that the specific format is not so significant as long as it has sufficient semantic content to be understandable. This results because the choice of element tag names, their sequence etc. is a personal preference: the power of XSL is always available for another user who wishes to use an alternative format. Thus arriving at a well annotated, self-defining and logical policy file format was the endeavor.

The XML User Policy file format we decided on is illustrated in the following example:

```
<Conditions>
  <ApplicationDomain app_domain="IPsec policy">
    <NetworkMode network_mode="normal">
      <SecurityLevel security_level="low">
          <Port local_filter_port="21"
                remote_filter_port="21">
            <Encapsulation esp_present="yes">
              <EncryptionAlgorithm esp_enc_alg="des" />
              <EncryptionAlgorithm esp_enc_alg="des3"/>
            </Encapsulation>
          </Port>
          <Port local_filter_port="23"
                remote_filter_port="23">
            <Encapsulation esp_present="yes">
              <EncryptionAlgorithm esp_enc_alg="des3"/>

              <EncryptionAlgorithm esp_enc_alg="aes" />
          </Encapsulation>
            <Ah ah_present="yes">
              <AuthenticationAlgorithm
                    ah_auth_alg="hmac-sha" />
              <AuthenticationAlgorithm
                    ah_auth_alg="des-mac" />
            </Ah>
          </Port>
        </SecurityLevel>
        <SecurityLevel security_level="medium">
          <Port local_filter_port="21"
                remote_filter_port="21">
            <Encapsulation esp_present="yes">
              <EncryptionAlgorithm esp_enc_alg="des3"/>
              <EncryptionAlgorithm
                    esp_enc_alg="des-iv32" />
            </Encapsulation>
          </Port>
        </SecurityLevel>
        <SecurityLevel security_level="high" />
      </NetworkMode>
      <NetworkMode network_mode="impacted">
        <SecurityLevel security_level="low" />
        <SecurityLevel security_level="medium" />
        <SecurityLevel security_level="high" />
      </NetworkMode>
      <NetworkMode network_mode="crisis">
        <SecurityLevel security_level="low" />
        <SecurityLevel security_level="medium" />
        <SecurityLevel security_level="high" />
      </NetworkMode>
    </ApplicationDomain>
  </Conditions>
  <Dummy><![CDATA[
    ]]></Dummy>
</Policy>
```

Having arrived at the XML policy file format, XSL stylesheets are used to transform the policy file into the desired formats. Two stylesheets were designed using XSLT (Refer Figure 1).

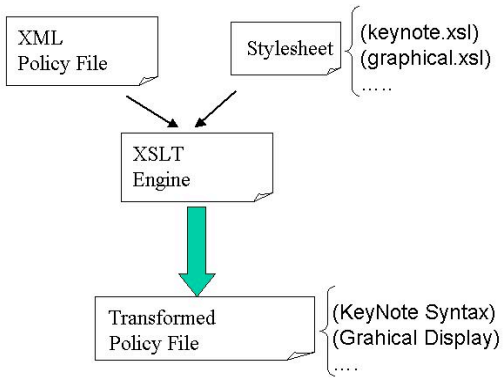

Figure 1. XSL transformation of XML

A stylesheet was created for transforming the file to the KeyNote policy file format. An alternative stylesheet to transform the XML policy file to a more human readable, graphical web based format was also written. The transformed output of the XML Policy file using this template is shown in Figure 6 (see Appendix).

5.1. Java-based GUI

Although XML provides us with the flexibility to edit the policy file in any XML editor, it would still be convenient to provide a graphical user interface to manipulate the policy file. This would help to eliminate inadvertent errors, and would enable global policy decisions to be applied throughout the policy file. An experienced system administrator could still capitalize on the use of the XML policy format and edit the file in the absence of the graphical user interface (GUI).

A Java-based GUI was therefore built to integrate various components of the software [9]. Drop down menus and dialog boxes guide the user to input various parameters required for the policy file. To enable maintenance of the GUI, called the *Policy-Editor*, a separate XML configuration file was used to feed the data for various drop down menus and combo/list boxes. This decoupling of the Java code from the configuration data will enable continued use of the Policy-Editor without the need to modify the Java code.

Figures 5 through 8 are screen shots of the Policy Editor. Figure 2 and Figure 3 select ports, and operational modes and security levels in the construction of a security policy. Figures 4 and 5 show the granular settings of encryption and authentication for particular ports. Figure 6 (see Appendix) shows how the XSL transformation of the resulting policy file displays the policy in a graphical and more intuitive format.

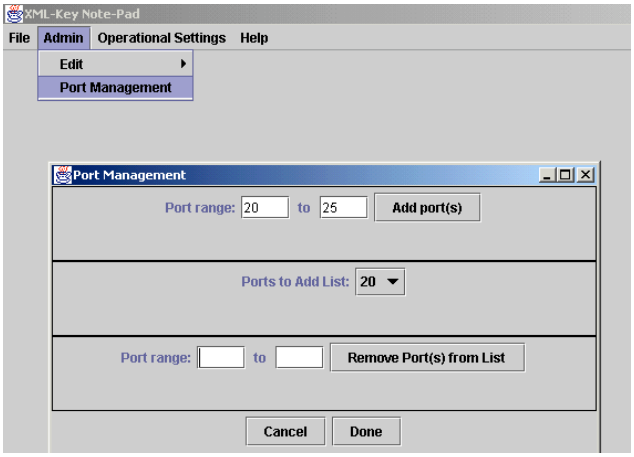

Figure 2. Managing ports in the admin module.

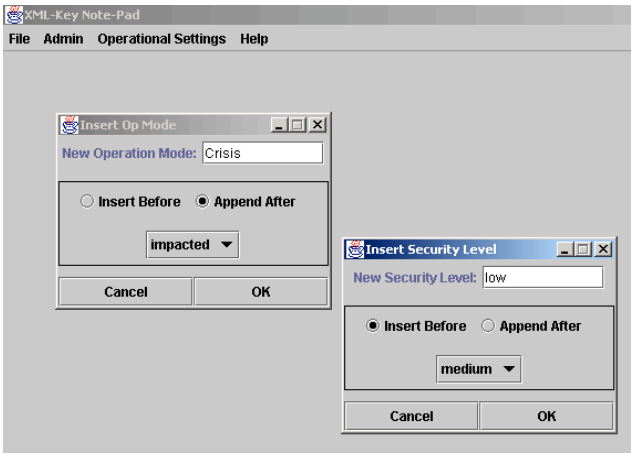

Figure 3. Admin mode settings for security level and op modes

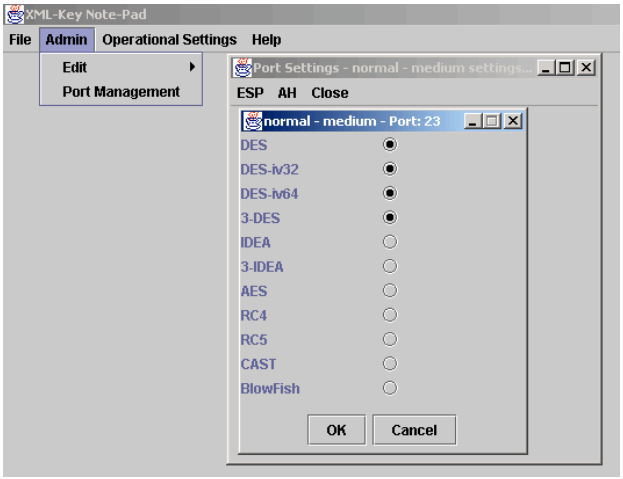

Figure 4. Encryption settings for ports

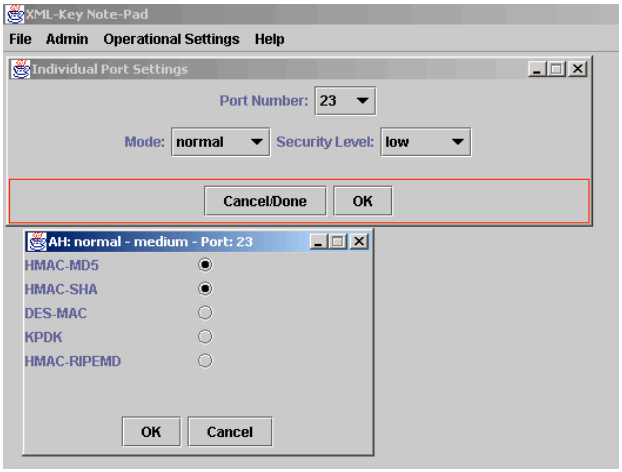

Figure 5. Authentication for AH mode

In addition to the policy editor a generic XML editor, such as XML Spy (Copyright ©1998-2002 Altova GmbH) can be used to view and edit an XML Policy file. Figure 7 (see Appendix) is a screen shot depicting the use of XML Spy editor to manipulate the XML Policy file directly. The result of schema validation can also be seen here. Figure 8 (see Appendix) is the design view of the schema when viewed in XML Spy, which allows specification and analysis of the policy from another viewpoint.

The security policy management toolkit is comprised of the Java based Policy-Editor and an XML editor such as XML Spy, XML Notepad, etc. The XML editors are not essential, but can aid in file manipulation, their transformation to multiple forms, and validation of schemas.

6. Future Work

Our work can be complemented with additional work in a number of areas to provide better tools for policy management. Listed below are several major items that will require attention.

6.1. Policy File Format

The XML policy file format currently specified could benefit from a more elaborate format with tags for other parameters. XML Namespaces and XML vocabularies could be utilized for a more comprehensive policy format [18]. Examples could involve incorporating other parameters such as algorithm key length, time-of-day parameters etc. The policy format should be able to accommodate other Boolean operators such as inequality definitions (<, >, !=) in the security policy management mechanism. For example, esp_enc_alg > DES could imply 3DES and AES if we have an ordering for the 'security strength' of each algorithm. Global

policy statements such as encryption in crisis mode < 3DES, etc., should be possible. Inclusion of IP addresses in policy statements should also be made possible. These concepts have been demonstrated in the implementation. Addition of more parameters as stated above could however open up possibilities for inconsistencies in policy statements and the same will have to be carefully and formally worked out.

6.2. Policy Editor Enhancements

The policy editor interface, though complete and functional, can be improved upon. The particular improvements envisioned are as follows:

6.2.1. Global policy settings. The policy editor could be modified to enable global policy settings. For instance we could have a statement such as all ports should have a minimum encryption of DES or the maximum encryption algorithm for Crisis mode should not exceed 3DES, etc. (again, assuming a partial ordering of such algorithms). The global settings option could enter the default settings for all permissible ports and then more granular changes could be made.

6.2.2. Translation and viewing XML. XML translation and viewing currently need the help of a general purpose XML editor. Using Java packages such as Javax, the same could be incorporated into the GUI thus dispensing the need for XML editors for translation.

6.2.3. Schema validation. Validation of the XML document against DTD and schema need to be incorporated into the GUI. Validation is currently performed using an XML tool such as XML Spy. DOM/JDOM/SAX could be used for this purpose.

6.2.4. Inconsistency and contradiction checks. As the policy file is extended to include global parameters and overlapping rules apply to a particular port or application, inconsistencies and contradictions could emerge. Various XML tools could help in avoiding this. Distributed IPsec policies could also give rise to policy consistency issues.

6.3. XML Interface to KeyNote.

Extending the XML policy language specified here to a broader XML specification and providing an XML processor in the KeyNote engine itself would greatly enhance the use of KeyNote. This could reduce the overhead of parsing in KeyNote and provide the power of XML for better auditing and dynamic management of trust. By providing an XML interface to KeyNote, application users could define their own versions of the policy language and use XSL for translating it into the desired KeyNote format, which would be trivial, or alternatively they could use the vocabulary specified in the KeyNote specifications.

7. Conclusion

Security policy management is a critical issue in the management of computer and networking resources. IPsec and KeyNote provide a mechanism to implement a granular security policy. Previous research in the area of 'Quality of Security Service' demonstrates how an adaptive security policy can provide enhanced security with optimal utilization of network resources. A missing link in this process was the difficulty in specifying a well-defined, granular, error free and consistent security policy in the language understood by the KeyNote trust management engine. We have presented a solution to this problem in the form of an easy to use yet powerful security policy editor. The work demonstrates that use of XML technology as a middle layer provides us with a means to combine the security of KeyNote with the simplicity of a policy editor. This novel approach also provides us all the benefits of XML, such as XSL and XML security. While XSL was extensively used, XML security tools could also be used in follow up future work.

References

[1] Agar Christopher, Dynamic Parameterization of IPsec, Master of Science Thesis, Department of Computer Science, Naval Postgraduate School, December 2001.

[2] Keromytis, A. D., Ioannidis, J. and Smith, J. M., *Implementing IPsec*, In Proceedings of the IEEE *Global Internet (GlobeCom) 1997,* pp. 1948 - 1952. November 1997, Phoenix, AZ.

[3] Blaze, M., Feigenbaum, J., Ioannidis, J., and Keromytis, A. D, *The KeyNote trust management system Version 2, (RFC 2704,* Network Working Group, September 1999, http://www.ietf.org/rfc/rfc2404.txt

[4] Blaze, M., Feigenbaum, J., and Keromytis, A. D., *KeyNote: Trust Management for Public-Key Infrastructures,* In Proceedings of the *1998 Security Protocols International Workshop,* Springer LNCS vol. 1550, pp. 59 - 63. April 1998, Cambridge, England. Also *AT&T Technical Report 98.11.1.*

[5] Blaze, M., Ioannidis, J. and Keromytis, A. D. *Trust Management and Network Security Protocols*, In Proceedings of the *1999 Security Protocols International Workshop,* April 1999, Cambridge, England.

[6] Blaze, M., Ioannidis, J. and Keromytis, A. D., *Trust Management for IPsec,* In Proceedings of the Internet Society *Symposium on Network and Distributed Systems Security (SNDSS) 2001,* pp. 139 - 151. February 2001, San Diego, CA.

[7] Hunter, D., Cagle, K., Dix, C., Kovack,R., Pinnock, J., and Rafter, J., *Beginning XML 2nd Edition* , Wrox Press Ltd,2002.

[8] Irvine, C.E. and Levin, T., *Quality of Security Service*, *Proceedings of the New Security Paradigms Workshop*, Cork, Ireland, September 2000

[9] Java 2 Standard Edition, V1.2.2 API Specification, http://java.sun.com/products/jdk/1.2/docs/api/, Sun Microsystems, Inc., 1999.

[10] Kent, S and Atkinson, R, *Security Architecture* for the Internet Protocol, RFC2401, Network Working Group, November 1998, http://www.ietf.org/rfc/rfc2401.txt

[11] Maughan, D., Schertler, M., Schneider M., Turner J., Internet Security Association and Key Management Protocol (ISAKMP), RFC 2408, Network Working Group, November 1998, http://www.ietf.org/rfc/rfc2409.txt

[12] Mohan, R. XML Based Adaptive Policy Mnagement in a trust management system Context, Masters Thesis, Naval Postgraduate School, Monterey, CA, September 2002.

[13] Spyropoulou, Evdoxia., Agar, Christopher D., Levin, Timothy, and Irvine, Cynthia, IPsec Modulation for the Quality of Security Service, NPS-CS-02-001, Naval Postgraduate School, January 2002.

[14] Spyropoulou, Evdoxia., Levin, Timothy, and Irvine, Cynthia, Demonstration of Quality of Security Service Awareness for IPsec, NPS-CS-02-003, Naval Postgraduate School, January 2002.

[15] Thayer, R., Doraswamy,N., and Glenn, R., IP Security Document Roadmap, RFC 2411, Network Working Group, November 1998, http://www.ietf.org/rfc/rfc2411.txt

[16] XML Specification, http://www.w3.org/TR/2000/REC-xml-20001006 , Aug 2002

[17] XML Schema specifications, http://www.w3.org/TR/xmlschema-0, Aug 2002

[18] XML Namespace Recommendation, http://www.w3.org/TR/REC-xml-names/

[19] XSLT Specifications, http://www.w3.org/TR/xslt, Aug 2002 RFC2396

Appendix: Detailed Screen Illustrations

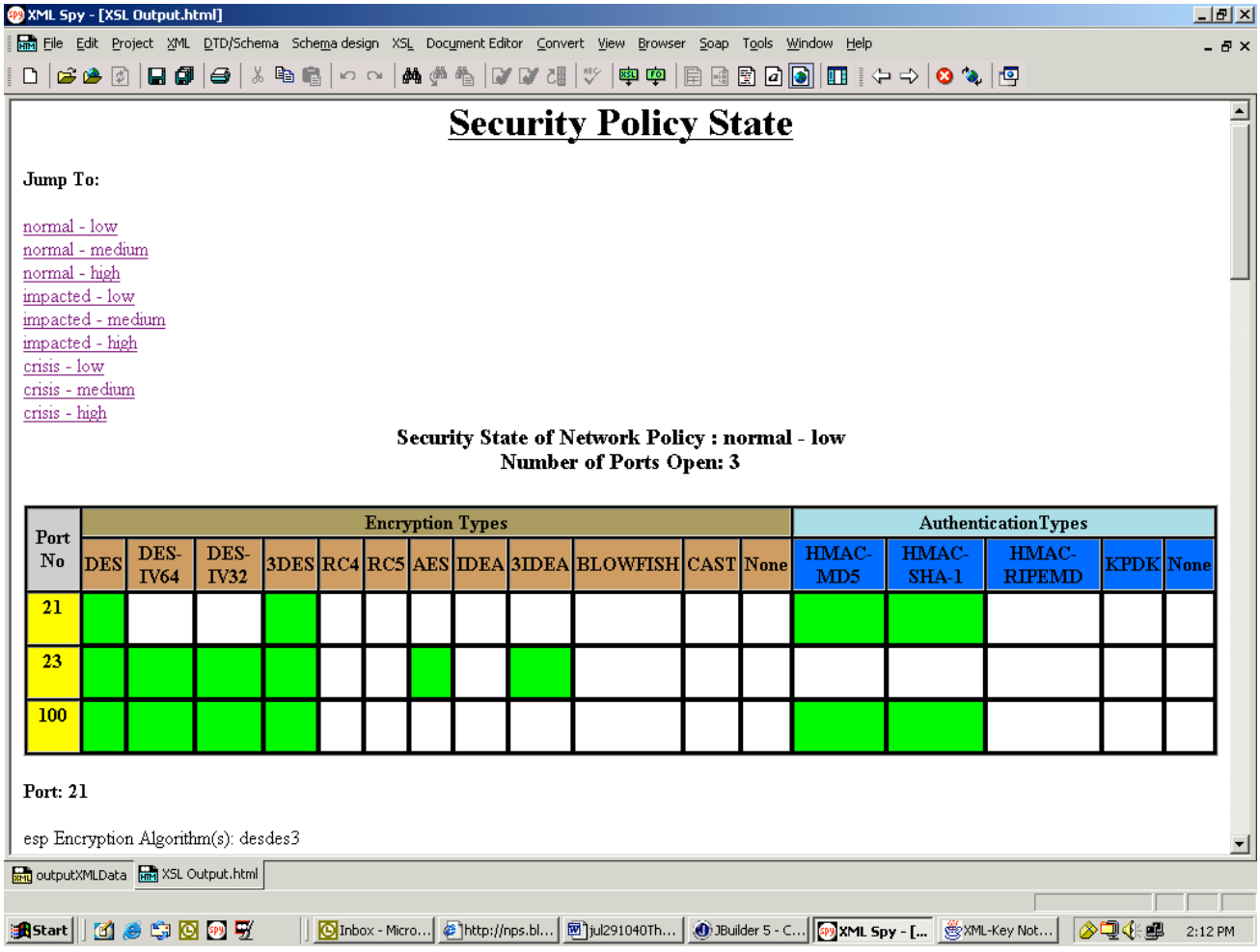

Figure 6. XSL transformation of the policy file

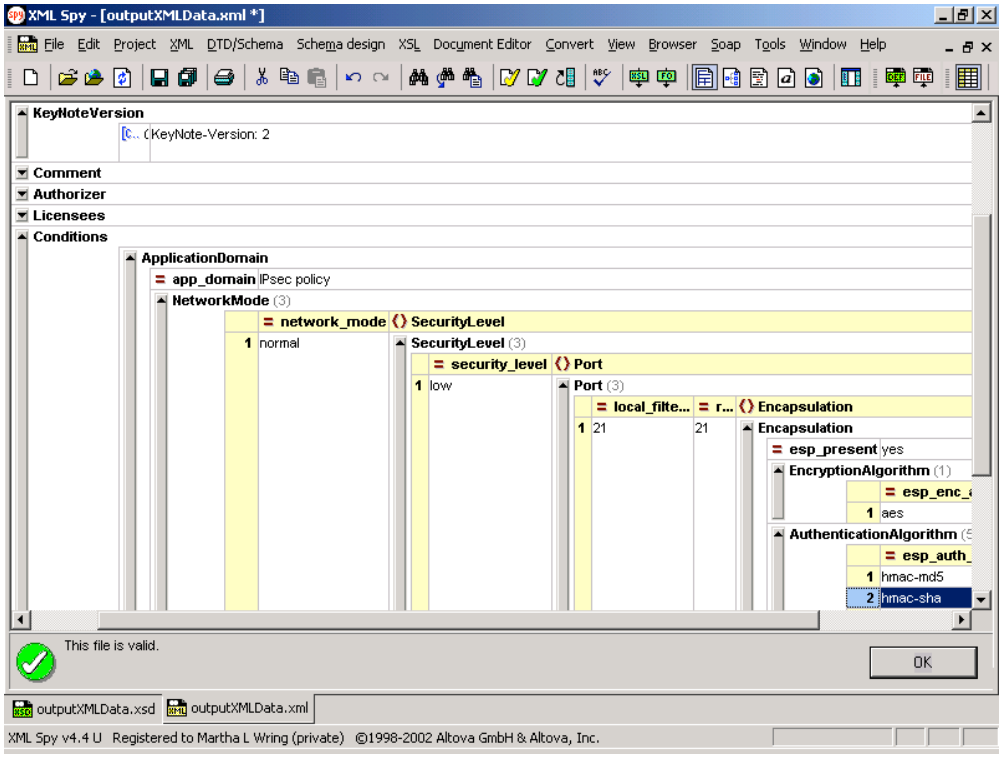

Figure 7. Editing and validation of XML policy file using XML Spy

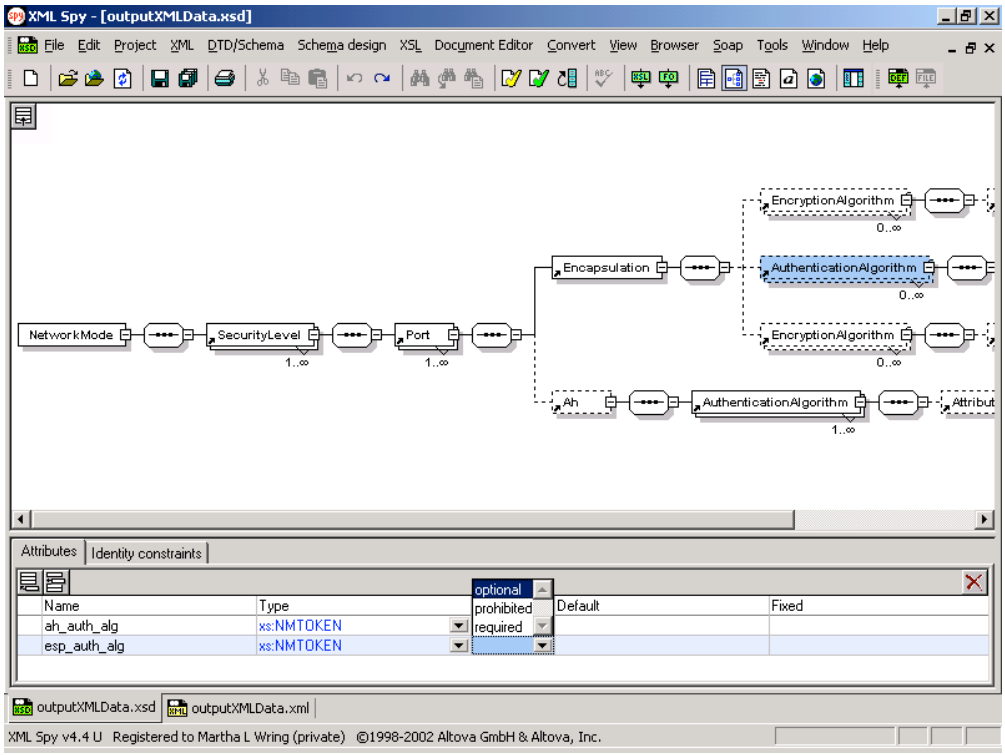

Figure 8. Schema design view of the XML policy document